Python Programing

- Pandas Cheatsheet
  - Cheatsheet 1
- Cheatsheet 2

Pandas Cheatsheet

Cheatsheet 1

In **pandas**, there are a few quick + reliable ways to check the **data type of columns**, depending on what you need.

1️⃣ Check *all* column types (most common)

```
df.dtypes
```

Output example

```
age         int64
height    float64
name       object
active       bool
date     datetime64[ns]
dtype: object
```

2️⃣ Detailed overview (very useful for debugging)

```
df.info()
```

Shows:

- column name
- non-null count
- dtype
- memory usage

3️⃣ Check type of *a single column*

```
df["age"].dtype
```

or

```
df.age.dtype
```

4️⃣ Check only numeric / non-numeric columns

```
df.select_dtypes(include="number").columns
```

```
df.select_dtypes(exclude="number").columns
```

5️⃣ Check if a column is a specific type

```
import pandas as pd

pd.api.types.is_numeric_dtype(df["age"])
pd.api.types.is_string_dtype(df["name"])
pd.api.types.is_datetime64_any_dtype(df["date"])
pd.api.types.is_bool_dtype(df["active"])
```

6️⃣ Count how many columns per dtype

```
df.dtypes.value_counts()
```

7️⃣ Detect mixed or problematic types (object columns)

```
df.select_dtypes(include="object").apply(
    lambda s: s.map(type).value_counts()
)
```

Useful when:

- CSV imported badly
- numbers stored as strings
- mixed types in one column

Task	Command
See all dtypes	`df.dtypes`
Full summary	`df.info()`
Single column	`df[col].dtype`
Only numeric	`select_dtypes()`
Type check	`pd.api.types.is_*`

Cheatsheet 2

Oke, ini **cheat-sheet Python (pandas)** buat **subset, grouping, dan filtering** — dari **paling dasar sampai yang sering dipakai di data real**. Contohnya aku sesuaikan dengan data kamu (provinsi, kab, gender, dll).

---

1 SUBSET (ambil sebagian data)

a) Subset kolom

```
df[["provinsi", "kab", "gender_ketua_kelompok"]]
```

b) Subset baris (index-based)

```
df.iloc[:10]       # 10 baris pertama
df.iloc[5:20]      # slice
```

```
df.loc[100:200]    # berdasarkan index label
```

c) Subset baris + kondisi

```
df[df["provinsi"] == "JAWA BARAT"]
```

2️⃣ FILTER (kondisi logika)

a) Filter satu kondisi

```
df[df["gender_ketua_kelompok"] == "L"]
```

b) Multiple kondisi (`&`, `|`)

```
df[
    (df["provinsi"] == "JAWA BARAT") &
    (df["gender_ketua_kelompok"] == "P")
]
```

⚠ WAJIB pakai tanda kurung!

---

c) Filter dengan `isin`

```
df[df["provinsi"].isin(["JAWA BARAT", "JAWA TENGAH"])]
```

d) Filter string (`contains`, `startswith`)

```
df[df["nama_kelompok"].str.contains("NELAYAN", case=False, na=False)]
```

---

e) Filter angka

```
df[df["skor_kelompok"] >= 80]
```

---

3 GROUPING (ringkas data)

a) Group by satu kolom

`df.groupby("provinsi").size()`

Sama dengan:

`df["provinsi"].value_counts()`

---

b) Group + agregasi

`df.groupby("provinsi")["nama_kelompok"].nunique()`

---

c) Group by multiple kolom

`df.groupby(["provinsi", "gender_ketua_kelompok"]).size()`

---

d) Banyak agregasi sekaligus

```
df.groupby("provinsi").agg(
    total_kelompok=("nama_kelompok", "count"),
    unik_kelompok=("nama_kelompok", "nunique"),
    rata_skor=("skor_kelompok", "mean"),
)
```

---

4 SUBSET + GROUP + FILTER (REAL CASE)

Jumlah kelompok per provinsi, hanya yang > 1000

```
(
    df.groupby("provinsi")
      .size()
      .reset_index(name="jumlah")
      .query("jumlah > 1000")
      .sort_values("jumlah", ascending=False)
)
```

Distribusi gender per provinsi

```
(
    df.groupby(["provinsi", "gender_ketua_kelompok"])
      .size()
      .unstack(fill_value=0)
)
```

Kelompok perempuan di Jawa Barat

```
df[
    (df["provinsi"] == "JAWA BARAT") &
    (df["gender_ketua_kelompok"] == "P")
][["nama_kelompok", "kab", "desa_kelompok"]]
```

5️⃣ FILTER DENGAN `query()` (lebih readable)

`df.query("provinsi == 'JAWA BARAT' and gender_ketua_kelompok == 'P'")`

Dengan `in`:

`df.query("provinsi in ['JAWA BARAT','JAWA TENGAH']")`

6️⃣ GROUP + FILTER (HAVING-style)

Contoh: **provinsi dengan ≥ 100 kelompok perempuan**

```
g = df[df["gender_ketua_kelompok"] == "P"]

(
    g.groupby("provinsi")
     .size()
     .loc[lambda s: s >= 100]
)
```

7️⃣ Best practice (penting banget)

✔ Jangan lupa normalisasi dulu

`df["provinsi"] = df["provinsi"].str.strip().str.upper()`
`df["gender_ketua_kelompok"] = df["gender_ketua_kelompok"].str.strip().str.upper()`

✔ Hindari chained indexing

`df[df["provinsi"] == "JAWA BARAT"]["kab"]`
✔

`df.loc[df["provinsi"] == "JAWA BARAT", "kab"]`

TL;DR (pola paling sering)

`df.loc[ df["kolom"].isin([...]), ["kolom1","kolom2"] ]`
`df.groupby([...]).agg(...)`
`df.query("kondisi")`